Analisis Performansi Topologi Jellyfish dengan Multipath TCP

Rafif Falih Sukmana Dalimunthe[1], Vera Suryani[2], Muhammad Arif Nugroho[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]rafiffalih@students.telkomuniversity.ac.id, [2]verasuryani@telkomuniversity.ac.id,
[3]arif.nugroho@telkomuniversity.ac.id

Abstrak

TCP yang berjalan pada transport layer umumnya digunakan untuk pengiriman paket data, akan tetapi jika TCP mengalami congestion Throughput yang dihasilkan tidak optimal. Pada data center Performansi diperlukan untuk menjaga kualitas layanan yang diberikan. Dalam penelitian ini membahas performansi topologi jellyfish pada data center menggunakan Multipath TCP (MPTCP). MPTCP dapat meningkatkan throughput dan memberikan performansi yang lebih baik. Yang menjadi parameter perbandingan adalah jumlah switch dan throughput. Dan algoritma MPTCP yang digunakan adalah LIA, OLIA, BALIA, dan wVegas. Hasil yang didapatkan pada penelitian ini didapatkan wVegas memiliki nilai throughput yang baik dalam skalabilitas switch dan skalabilitas host. Dikarenakan pada wVegas menggunakan delay sebagai congestion signal berbeda dengan LIA, OLIA, dan BALIA yang menggunakan packet loss sebagai congestion signal.

Kata kunci : data center, jellyfish, multipath tcp, linked increase alhorithm, opportunistic linked increases algorithm, balanced linked adaptation, weighted vegas.

Abstract

TCP which runs at the transport layer is generally used for sending data packets, but if TCP experiences congestion the resulting throughput is not optimal. In the data center, performance is needed to maintain the quality of the services provided. In this study, it discusses the performance of jellyfish topology in the data center using Multipath TCP (MPTCP). MPTCP can increase throughput and provide better performance. The parameters for comparison are the number of switches and the throughput. And the MPTCP algorithm used is LIA, OLIA, BALIA, and wVegas. The results obtained in this study show that wVegas has a good throughput value in switch scalability and host scalability. Because wVegas uses delay as a congestion signal, it is different from LIA, OLIA, and BALIA which use packet loss as a congestion signal. Keywords:

data center, jellyfish, multipath tcp, linked increase alhorithm, opportunistic linked increases algorithm, balanced linked adaptation, weighted vega

1. Pendahuluan

Latar Belakang

Dengan adanya migrasi daya komputasi dan aplikasi dari server lokal ke cloud, data center saat ini memiliki peranan penting bagi jaringan lokal sampai internet[19]. Karena data center sebagai pondasi dari sebuah infrastruktur jaringan. Dengan perkembangan teknologi yang begitu pesat, data center kini memiliki banyak komputer yang saling terhubung dengan kebutuhan bandwidth yang cukup besar[3]. Jellyfish merupakan sebuah topologi yang memiliki jaringan interkoneksi yang sangat tinggi yang mengimplementasikan random graph. Jellyfish adalah topologi degree-bounded random graph diantara top-of-rack (ToR) switch.

Data Center mempunyai peran yang krusial pada peningkatan service dan big data. Tingkat efesiensi Data Center memerlukan kapasistas jaringan yang besar dalam hal throughput. Protokol TCP yang menggunakan satu jalur masih memiliki kekurangan lain yaitu pada congestion jika berlebihan akan menyebabkan packet loss. Saat packet loss terjadi, packet sender akan menghentikan mengirim data hingga data yang hilang berhasil di transmisikan ulang dan throughput menjadi tidak optimal karena terdapat packet loss yang berlebihan[7].

Multipath TCP memungkinkan koneksi TCP untuk beroperasi secara bersamaan dengan beberapa antarmuka yang ada untuk meningkatkan pemanfaatan sumber daya ketangguhan koneksi. Multipath TCP meningkatkan bandwidth dan menciptakan subflow secara dinamis, subflow menunjukkan bahwa Multipath TCP dapat mencapai tiga kali throughput dari tcp. TCP data akan dibahi ke TCP subflow di Multipath TCP untuk ditransfer dalam jaringan pada waktu yang sama[5].

Multipath TCP dapat digunakan saat data center dikhususkan untuk menjalankan sistem terdistribusi salah satu contohnya adalah pemrograman parallel[12]. Dengan multipath tcp pengiriman data dapat lebih cepat dikarenakan packet loss akan berkurang dan pengiriman data dapat dilakukan menggunakan dua atau lebih subflow.

Topik dan Batasannya

Topik yang diangkat dalam penelitian ini meliputi analisis perbandingan *Multipath TCP* (MPTCP) dalam topologi *Jellyfish* di *Data Center*. Berdasarkan topik tersebut, terdapat beberapa batasan masalah yaitu:

- Topologi dibangun di*emulator Mininet*
- Setiap *link* antar *Switch* diberikan *Bandwidth* sebesar 1000 Mbps.
- *Controller* yang digunakan adalah *POX*.
- Algoritma *MPTCP* yang digunakan adalah *LIA, OLIA, BALIA,* dan *wVegas*
- Koneksi yang dibuat adalah koneksi inter *Data Center* dalam satu jaringan

Tujuan

Berikut adalah tujuan yang ingin dicapai pada Tugas Akhir.

- Untuk mengetahui bagaimana kinerja mptcp pada topologi jellyfish di data center dengan parameter jumlah switch, throughput, dan skalability;
- Untuk mengetahui algoritma LIA, BALIA, OLIA, atau wVegas yang mempunyai throughput terbesar dan nilai fairness baik pada topologi jellyfish.

Organisasi Tulisan

Penelitian ini ditulis menjadi beberapa bab. Pada bab pertama akan menjelaskan tentang abstrak, latar belakang, rumusan masalah, dan tujuan dari topik yang dikerjakan. Pada bab kedua akan membahas tentang penelitian yang berkaitan dengan topik yang diangkat. Skenario pengujian dan metode yang digunakan pada penelitian ini akan dijelaskan pada bab 3. Bab 4 akan menjelaskan tentang hasil evaluasi dan pada bab 5 akan menjelaskan kesimpulan.

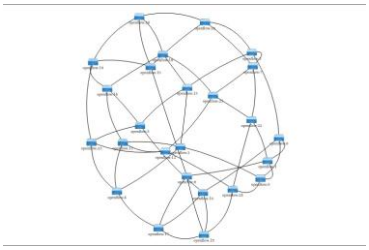
2. Studi Terkait

2.1 Topologi Data Center

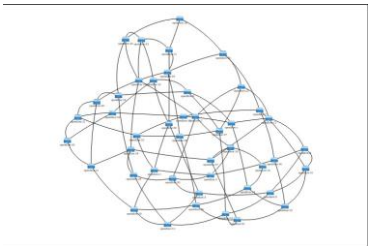
Data Center merupakan salah satu fasilitas yang menjadi pondasi infrastruktur jaringan. Dengan perkembangan teknologi yang sangat cepat data center menjadi jantung dalam berbagai bidang karena data center saat ini menjadi pusat pertukaran data dan komputasi data dengan skala yang cukup besar. Oleh karena itu arsitektur dibangun tiga perhitungan yaitu bandwidth, latency, dan realibility. Dengan bandwidth yang cukup besar dan latency yang kecil, data center tidak bisa lepas dari kerusakan sistem.

2.1.1 Jellyfish

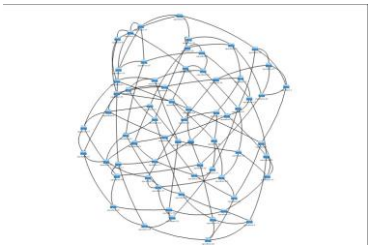
Jellyfish adalah sebuah topologi yang memiliki jaringan interkoneksi yang sangat tinggi yang mengimplementasikan random graph. Dibandingkan dengan topologi Fat-Tree, Jellyfish memiliki efisiensi biaya yang lebih baik dibandingkan dengan Fat-Tree[14]. Selain itu, Jellyfish memiliki nilai throughput yang lebih baik dibandingkan dengan beberapa topologi data center lainnya[13]. Berdasarkan[15] dalam kasus sederhana setiap switch memiliki jumlah port yang sama yaitu $k = k_i$ dan $r = r_i$ dimana k adalah jumlah port switch dan r adalah port switch yang digunakan untuk menghubungkan top-of-rack (ToR) switch dimana Top-of-rack switch adalah switch yang berada pada rak server untuk menghubungkan peralatan komputasi seperti server. Dengan N adalah rak server, jaringan tersebut mendukung server sebanyak $N(k-r)$. Hal tersebut dapat dinotasikan menggunakan notasi Random Regular Graph yaitu RRG(N,k,r). Random Regular Graph adalah graph yang menunjukkan ruang probabilitas pada semua graph. Berikut ini adalah contoh topologi Jellyfish dengan jumlah switch 25 switch, 45 switch dan 65 switch tiap switch memiliki 6 port serta drajat yang digunakan adalah empat.



Gambar 1. Topologi Jellyfish 25 switch



Gambar 2. Topologi Jellyfish 45 switch



Gambar 3. Topologi Jellyfish 65 switch

2.2 Multipath TCP

Multipath TCP adalah sebuah ekstensi dari protokol TCP untuk menggunakan beberapa IP sebagai jalur transportasi dalam jaringan. Hal ini membuat kecepatan dan kehandalan transfer meningkat[10]. Protocol MPTCP bekerja pada transport layer sama seperti TCP. Perbedaannya dapat dilihat pada protocol stack antara TCP dan MPTCP

Application
TCP
IP

Gambar 4. Protocol Stack TCP

Application	
MPTCP	
Subflow (TCP)	Subflow (TCP)
IP	IP

Gambar 5. Protocol Stack MPTCP

Pada gambar 4 menunjukkan TCP hanya menggunakan satu IP address sebagai identitas satu host, sehingga host hanya menggunakan satu network interface. Pada gambar 5 menunjukkan protocol stack MPTCP, disini dapat dilihat bahwa MPTCP memiliki lebih dari satu IP address yang digunakan untuk berkomunikasi. MPTCP connection merupakan sekumpulan subflow yang digunakan untuk komunikasi antar host. Subflow pada MPTCP merupakan flow dari TCP segment yang bekerja menggunakan satu individual path.

2.3 Linked Increases Algorithm

Linked Increase Algorithm (LIA) [6], mengkalikan *additive increase function* dari *subflows* yang berbeda dengan prinsip *resource pooling* dan menggunakan *standard TCP behaviour* jika terjadi *packet loss*. Algoritma ini memungkinkan pengalihan traffic dari jalur yang padat ke yang tidak padat. Implementasi kernel linux menggunakan algoritma ini sebagai default.

2.3.1 Opportunistic Linked Increases Algorithm

Opportunistic Linked-Increases Algorithm (OLIA) [9], seperti *LIA* akan mengkalikan *additive increase* dan menggunakan standard *TCP behavior* jika terjadi *drop packet loss*, tetapi berbeda dalam bagian peningkatan sesuai dengan *window increment* sebagai fungsi dari *RTTs*

2.3.2 Balanced Linked Adaptation

Balanced Linked Adaptation Algorithm (BALIA) [17], LIA dan OLIA memiliki kekurangan yang tidak mendukung terhadap Single Path TCP (SPTCP) atau tidak responsif terhadap perubahan network pada kondisi tertentu. BALIA hampir sama dengan algoritma sebelumnya dengan memiliki balance yang bagus antara TCP-friendliness, responsiveness, dan window oscillation.

2.3.3 Weighted Vegas

Delay-based Algorithm atau Weighted Vegas (wVegas)[18], adalah algoritma *congestion control* adopsi dari TCP vegas yang menggunakan *queueing delay* sebagai *congestion signal*. *wVegas* dapat menyediakan *fair* dan *effecient traffic shifting* yang artinya setiap *flow* akan semaksimal mungkin untuk menyamakan tingkat congestion pada semua *available path*, prinsip ini disebut juga *congestion equality principle*.

2.4 Software Defined Network

Software Defined Network atau yang biasa disingkat SDN adalah suatu paradigma baru dalam dunia jaringan komputer yang memisahkan control plane dan data plane [16]. Pemisahan ini bertujuan untuk manajemen jaringan yang terpusat dan mudah dirawat oleh administrator jaringan. Untuk mencapai tujuan ini dibuatkan sebuah protokol baru yaitu OpenFlow yang digunakan untuk komunikasi antara controller dan perangkat jaringan.

Peran Controller dalam SDN adalah sebagai pusat pengendali logik yang menerima instruksi dari lapisan aplikasi dan meneruskannya ke lapisan infrastruktur[2]. Controller mengirimkan informasi ke data plane menggunakan protokol OpenFlow.

SDN pada penelitian ini duganakan sebagai alat bantu untuk membangun topologi Jellyfish.

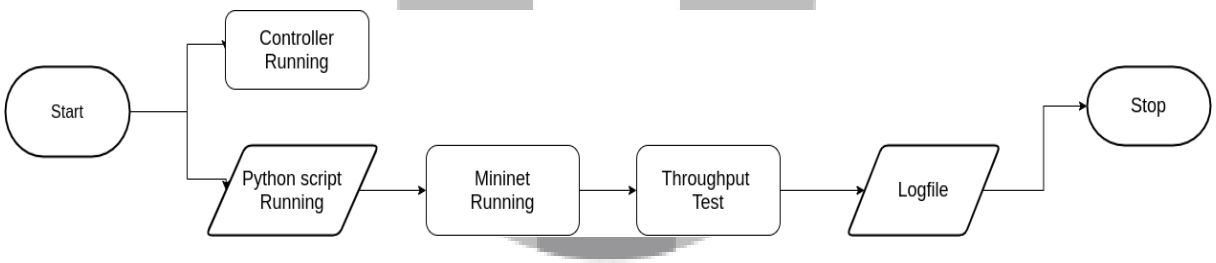
2.5 Mininet

Mininet adalah sebuah emulator yang biasa digunakan pada sistem operasi Linux. Emulator digunakan untuk membuat sebuah topologi jaringan secara virtual yang terdiri dari host, switch, controller, serta link. Mininet merupakan emulator yang mendukung OpenFlow dan biasa digunakan untuk melakukan emulasi terhadap topologi jaringan dengan arsitektur jaringan Software Defined Network[1].

Untuk membuat sebuah topologi jaringan pada mininet, dapat dilakukan dengan melakukan modifikasi pada code yang telah disediakan oleh Mininet. Atau dengan membuat sebuah file berbasis Python. Pada mininet, dapat dilakukan testing terhadap jaringan dengan topologi yang kompleks dan melakukan analisis terhadap bandwidth, konektifitas, dan kecepatan.

3. Perancangan Sistem

3.1 Skema Pengujian



Gambar 6. Skema Pengujian

Dapat dilihat pada Gambar 6 Skema pengujian dilakukan dengan menjalankan script python yang akan mengotomatis membuat topologi Jellyfish dengan dengan argumen yang diberikan dan menghubungkan mininet dengan controller. Script python akan langsung menlakukan pengujian Throughput, hasil kedua pengujian tersebut akan disimpan dalam bentuk format file txt.

3.2 Skenario Simulasi

Skenario Simulasi dibagi menjadi 2 skenario pada metode LIA, OLIA, BALIA, dan wVegas dengan parameter pengukuran yaitu Jumlah host dan Throughput.

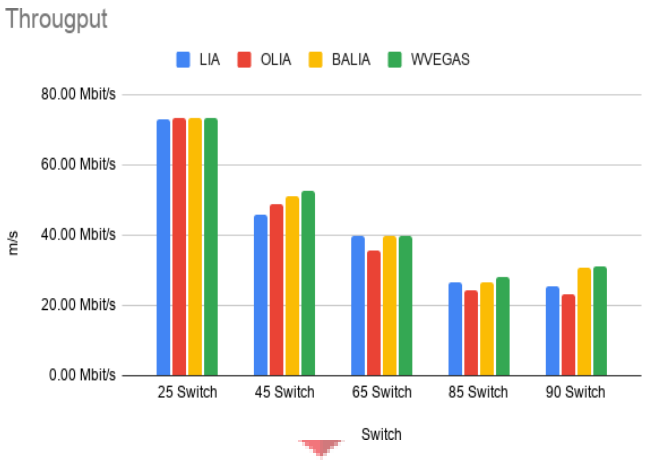
Jumlah host disini digunakan sebagai parameter untuk skalabilitas, karena data center pasti akan berkembang sehingga besar kemungkinannya akan ada penambahan jumlah switch. Dengan pengujian membuat topologi Jellyfish dan dibuat satu host1 untuk menjadi iperf server dan secara serentak semua host selain host1 akan melakukan test ke host1 dengan menaikkan skala jumlah hostnya dengan jumlah switch tetap pada 20 switch.

Throughput adalah faktor utama pada data center, karena dari throughput kecepatan data dikirim dapat dilihat seberapa cepat pengirimannya. Dalam penelitian ini akan menggunakan aplikasi iperf sebagai pengukur throughput. Dan akan dibuat satu host1 untuk menjadi iperf server dan secara serentak semua host selain host1 akan melakukan test ke host1.

4. Evaluasi

4.1 Hasil Pengujian Throughput

Berikut adalah hasil rata-rata *Throughput* yang didapatkan setelah dilakukan lima kali percobaan.



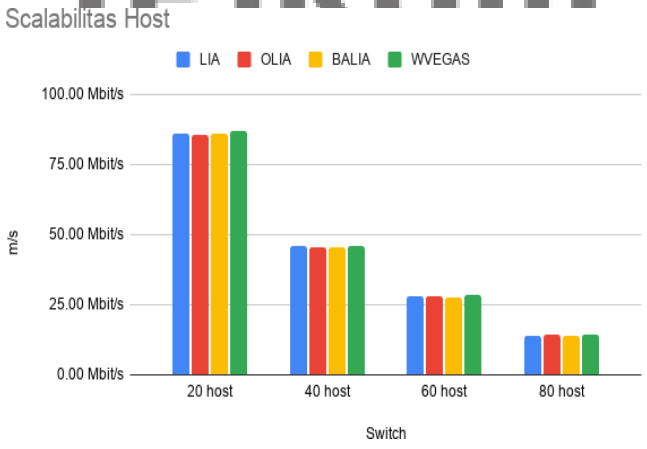
Gambar 7. Grafik Throughput

Tabel 1. **Hasil Pengujian Throughput**

Congestion Control	25 Switch	45 Switch	65 Switch	85 Switch	90 Switch
LIA	73.09 Mbps	45.64 Mbps	39.79 Mbps	26.46 Mbps	25.30 Mbps
OLIA	73.16 Mbps	48.66 Mbps	35.74 Mbps	24.39 Mbps	23.18 Mbps
BALIA	73.20 Mbps	50.89 Mbps	39.73 Mbps	26.57 Mbps	30.66 Mbps
WVEGAS	73.26 Mbps	52.46 Mbps	39.58 Mbps	27.92 Mbps	31.04 Mbps

4.2 Hasil Pengujian Scalability

Berikut adalah hasil rata-rata *Scalability* yang didapatkan setelah dilakukan lima kali percobaan.



Gambar 8. Grafik Scalability

Tabel 2. **Hasil Pengujian Scalability**

Congestion Control	20 Host	40 Host	60 Host	80 Host
LIA	85.95 Mbps	45.85 Mbps	28.20 Mbps	13.79 Mbps
OLIA	85.62 Mbps	45.62 Mbps	27.85 Mbps	14.16 Mbps
BALIA	85.88 Mbps	45.59 Mbps	27.73 Mbps	13.85 Mbps
WVEGAS	86.80 Mbps	46.08 Mbps	28.54 Mbps	14.26 Mbps

4.3 Pengujian Fairness

Pada pengujian fairness digunakan sebuah rumus untuk mengetahui apakah masing masing host mendapatkan throughput yang setara. Rumus yang digunakan adalah rumus Raj Jain [8]

$$\frac{(\sum x_i)^2}{n \cdot \sum x_i^2}$$

Dimana *x* adalah throughput dan *n* adalah jumlah host yang digunakan. Dengan rumus ini akan dihitung fairness yang didapatkan pada skala 40 host pada tiap tiap Congestion Control dan didapatkan untuk hasilnya sebagai berikut.

Tabel 3. **Hasil Pengujian Fairness**

Congestion Control	Hasil Fairness
LIA	0.8070
OLIA	0.8235
BALIA	0.8469
WVEGAS	0.8336

4.4 Analisis Hasil Pengujian

Berdasarkan Tabel 1 terlihat bahwa *wVegas* memiliki nilai rata rata Bandwidth yang lebih tinggi dibandingkan dengan metode yang lain. Dikarenakan *WVegas* berbeda dari *LIA*, *OLIA*, dan *BALIA* yang menggunakan *packet loss* sebagai *congestion signal* sementara *wVegas* menggunakan *delay* sebagai *congestion signal* karena itu *wVegas* lebih sensitif terhadap perubahan *traffic*[4]. Karena *wVegas* menggunakan *delay* sebagai *congestion signal* maka didapatkanlah penyeimbangan beban yang baik. *WVegas* menerapkan *wwight* pada tiap subflow dan akan menyesuaikan sesuai dengan prinsip *Congestion Equality*. Bobot yang diberikan pada tiap subflow akan mengukur *Bandwidth* pada *subflow* tersebut. Dengan demikian subflow yang tidak terlalu padat akan memperoleh bobot tambahan.

Berdasarkan Tabel 2 dapat dilihat *Wvegas* memiliki nilai terbaik dalam setiap skalabilitas host, dikarenakan *Wvegas* melakukan trafic shifting dengan *delay* sebagai congestion signalnya dan akan membagi beban pada tiap subflow setara.

Fairness adalah salah satu goal dari dibuatnya Multipath TCP seperti pada dokumen RFC[11]. Berdasarkan Tabel 3 Dapat dilihat *BALIA* memiliki nilai fairness mendekati 1 oleh karena itu *BALIA* membagi flow TCP paling adil dibandingkan congestion control MPTCP lainnya. Ini dikarenakan tujuan BALIA diciptakan adalah meningkatkan friendliness yang kurang pada OLIA.

5. Kesimpulan

5.1 Kesimpulan

Berdasarkan hasil analisis pengujian yang telah dilakukan, maka dapat disimpulkan bahwa:

1. Jumlah *switch* pada *Jellyfish* sangat berpengaruh dalam mendapatkan jalur dengan *cost* yang sama karena semakin sedikit jumlah switch kemungkinan jalur dengan *cost* yang sama akan sulit ditemukan dikarenakan *Jellyfish* adalah topologi berbasis *random graph*.

2. *wVegas* memiliki nilai terbaik dalam *throughput* dan *Skalabilitas* pada topologi *Jellyfish*. Sementara *BALIA* memiliki nilai Fairness yang paling baik diantara congestion control multipath tcp yang lain

5.2 Saran

Saran yang dapat diberikan untuk pengembangan penelitian berikutnya adalah:

1.Menggunakan *topologi* data center lainnya seperti Fat-Tree atau Xpander

2.Menambahkan *background traffic* di dalam pengujian.

Reference

[1] Mininet description. http://mininet.org. Accessed: 2019-10-24.

[2] Understanding the sdn architecture - sdn control plane sdn data plane. https://www.sdxcentral.com/networking/sdn/definitions/inside-sdn-architecture/. Accessed: 2019-10-24.

[3]M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 63–74. ACM, 2008.

[4]Y. Cao, M. Xu, and X. Fu. Delay-based congestion control for multipath tcp. In *2012 20th IEEE International Conference on Network Protocols (ICNP)*, pages 1–10. IEEE, 2012.

[5]R. K. Chaturvedi and S. Chand. Mptcp over datacenter network. pages 894–898, 2018.

[6]M. Handley, C. Raiciu, and D. Wischik. Coupled congestion control for multipath transport protocols. 2011.

[7]J. Hwang, A. Walid, and J. Yoo. Fast coupled retransmission for multipath tcp in data center networks. *IEEE Systems Journal*, 12(1):1056–1059, 2016.

[8]R. K. Jain, D.-M. W. Chiu, W. R. Hawe, et al. A quantitative measure of fairness and discrimination. *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, 1984.

[9]R. Khalili, N. Gast, M. Popovic, et al. Opportunistic linked-increases congestion control algorithm for mptcp. 2013.

[10]B. Y. L. Kimura and A. A. F. Loureiro. MPTCP linux kernel congestion controls. *CoRR*, abs/1812.03210, 2018.

[11]C. Raiciu, M. Handley, and D. Wischik. Coupled congestion control for multipath transport protocols. Technical report, IETF RFC 6356, Oct, 2011.

[12]O. Segal, M. Margala, S. R. Chalamalasetti, and M. Wright. High level programming framework for fpgas in the data center. In *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4. IEEE, 2014.

[13]A. Singla, P. B. Godfrey, and A. Kolla. High throughput data center topology design. In *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, pages 29–41, 2014.

[14]A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey. Jellyfish: Networking data centers randomly. In *Presented as part of the 9th{ USENIX} Symposium on Networked Systems Design and Implementation ({ NSDI} 12)*, pages 225–238, 2012.

[15]A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey. Jellyfish: Networking data centers randomly. In *Presented as part of the 9th{ USENIX} Symposium on Networked Systems Design and Implementation ({ NSDI} 12)*, pages 225–238, 2012.

[16]B. Underdahl and G. Kinghorn. *Software Defined Networking for Dummies*. John Wiley Sons, Inc, 2015.

[17]A. Walid, Q. Peng, J. Hwang, and S. Low. Balanced linked adaptation congestion control algorithm for mptcp. *Working Draft, IETF Secretariat, Internet-Draft draft-walid-mptcp-congestion-control-04*, 2016.

[18]M. Xu, Y. Cao, and E. Dong. Delay-based congestion control for mptcp. *IETF, work in progress, Internet-draft draft-xu-mptcpcongestion-control-01*, 2015.

[19]U. Zakia and H. B. Yedder. Dynamic load balancing in sdn-based data center networks. In *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 242–247. IEEE, 2017.

Lampiran

```
------------------------------------------------------------
Client connecting to 10.0.0.1, TCP port 5001
TCP window size: 86.2 KByte (default)
------------------------------------------------------------
[127] local 10.0.0.2 port 51248 connected with 10.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[127]  0.0- 1.0 sec  40.2 MBytes   338 Mbits/sec
[127]  1.0- 2.0 sec  19.9 MBytes   167 Mbits/sec
[127]  2.0- 3.0 sec  5.00 MBytes  41.9 Mbits/sec
[127]  3.0- 4.0 sec  4.50 MBytes  37.7 Mbits/sec
[127]  4.0- 5.0 sec  3.38 MBytes  28.3 Mbits/sec
[127]  5.0- 6.0 sec  6.62 MBytes  55.6 Mbits/sec
[127]  6.0- 7.0 sec  4.00 MBytes  33.6 Mbits/sec
[127]  7.0- 8.0 sec  4.00 MBytes  33.6 Mbits/sec
[127]  8.0- 9.0 sec  3.88 MBytes  32.5 Mbits/sec
[127]  9.0-10.0 sec  1.89 MBytes  15.8 Mbits/sec
[127] 10.0-11.0 sec  1.25 MBytes  10.5 Mbits/sec
[127] 11.0-12.0 sec  1.25 MBytes  10.5 Mbits/sec
[127] 12.0-13.0 sec  1.25 MBytes  10.5 Mbits/sec
[127] 13.0-14.0 sec  1.22 MBytes  10.3 Mbits/sec
[127] 14.0-15.0 sec  1.25 MBytes  10.5 Mbits/sec
[127] 15.0-16.0 sec  10.4 MBytes  87.0 Mbits/sec
[127] 16.0-17.0 sec  4.75 MBytes  39.8 Mbits/sec
[127] 17.0-18.0 sec  2.75 MBytes  23.1 Mbits/sec
[127] 18.0-19.0 sec  1.38 MBytes  11.5 Mbits/sec
[127] 19.0-20.0 sec  2.00 MBytes  16.8 Mbits/sec
[127] 20.0-21.0 sec  1.24 MBytes  10.4 Mbits/sec
[127] 21.0-22.0 sec  1.24 MBytes  10.4 Mbits/sec
[127] 22.0-23.0 sec  1.18 MBytes  9.90 Mbits/sec
[127] 23.0-24.0 sec  1.24 MBytes  10.4 Mbits/sec
```

Gambar 9. Hasil log iperf